NPS-55Ss74071

NAVAL POSTGRADUATE SCHOOL

Monterey, California

ANALYSIS OF ERROR PROCESSES

IN COMPUTER SOFTWARE

by

Norman F. Schneidewind

July 1974

Approved for public release; distribution unlimited.
Prepared For: Naval Electronics Laboratory Center, San Diego, California and Navy Fleet Material Support Office, Mechanicsburg, Pa.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Isham Linder
Superintendent

Jack R. Borsting
Provost

The work reported herein was supported in part by the Naval Electronics Laboratory Center (NELC), Office of Naval Research and the Fleet Material Support Office.

This report was prepared by:

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NPS-55Ss74071
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): Analysis of Error Processes in Computer Software

5. TYPE OF REPORT & PERIOD COVERED: Technical report

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Norman F. Schneidewind

8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NELC: Work Request 2-9049; FMSO: Proj. Order 3-3508

11. CONTROLLING OFFICE NAME AND ADDRESS: NELC, Office of Naval Research, Fleet Material Support Office

12. REPORT DATE: July 1974

13. NUMBER OF PAGES: 40

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software measurement
Software reliability
Software error analysis
Software quality control
Stochastic processes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A non-homogenous Poisson process is used to model the occurrence of errors which occur during functional testing of command and control software. The parameters of the detection process are estimated by using a combination of maximum likelihood and weighted least squares methods. Once parameter estimates are obtained, forecasts can be made of cumulative number of detected errors. Forecasting equations of cumulative corrected errors, errors detected but not corrected, and the time required to detect or correct a

(cont. next page)

specified number of errors, are derived from the detected-error function. The various forecasts provide decision aids for managing software testing activities. Naval Tactical Data System software error data are used to evaluate several variations of the forecasting methodology and to test the accuracy of the forecasting equations. Because of changes which take place in the actual detected error process, it was found that recent error observations are more representative of future error occurrences than are early observations. Based on a limited test of the model, acceptable accuracy was obtained when using the preferred forecasting method.

INTRODUCTION

The purpose of this paper is to describe a model for statistically analyzing software error detection and correction processes during software functional testing. The purpose of the model is to provide decision aids for controlling the quality of command and control system software. The inputs to the model are error detection histories and the outputs are forecasts of the future behavior of error detection and correction processes. The model outputs would provide software production and quality control management with quantitative guidelines for:

- establishing testing strategies,
- making the acceptance/rejection decision,
- evaluating the tradeoff between incremental quality improvement and incremental resource investment.

SCOPE

The scope of this paper is to define certain software error terminology, set forth the mathematical formulation of the model and focus in detail on the methodology used for error detection and correction forecasting. With respect to the last item, forecasted values are compared with actual counts of detected and corrected errors obtained from a limited number of Navy Tactical Data System (NTDS) software trouble reports. Several variations of the forecasting methodology are compared and tentative conclusions are drawn concerning the validity of the methodology. The validation of the forecasting methodology against a large amount of NTDS software error data was beyond the scope of this particular research effort.

TERMINOLOGY

Increasingly, the quantitative evaluation of computer software is recognized as critically important to the effective functioning of computer systems. The body of knowledge, literature and models concerned with the measurement and evaluation of software characteristics is growing [1, 2, 3, 4, 5, 6]. Despite this upsurge in activity, there do not exist accepted definitions and methodology for describing and analyzing software error measurements. Therefore, it is inappropriate to use such terms as "software error," "software quality control" or "software quality assurance" with the expectation of achieving uniformity of interpretation and acceptance. Consequently, certain terms will be defined as they apply to this paper.

- Software Error. Here we are concerned with errors in programming logic which lead to undesirable results during program execution. Examples are the storing of data in incorrect memory locations or accessing the wrong file on a disc unit. Deviations from performance which are caused by an inherent limitation of the numerical or algorithmic technique are called performance deficiencies and are not classified as software errors. An example is the lack of precision in a computation which results from the truncation property of the algorithm, or an insufficient word length in the hardware. Also, errors or failures which are strictly attributable to hardware are not counted as software errors.
- Software Quality. This is the propensity of errors to occur in a program under stated operating conditions, as determined by such measures as the number of errors per unit time, time between errors, errors per instruction executed and criticality of certain errors to mission success.

- Software Quality Control. Management systems and procedures which are employed during program production and testing to produce software with acceptable error characteristics.
- Functional Testing. A type of software testing designed to ensure the user that system functions, such as target tracking, can be executed correctly. This type of testing is normally conducted by the user after the individual modules have been debugged by programmers and the modules have been integrated together as a program.
- Software Quality Assurance. Criteria for user acceptance or rejection of software products and the user acceptance tests (tests of user functions) and procedures used during functional testing.
- Modules. A set of computer instructions which accomplishes some specific function, such as the tracking or display function in NTDS.
- Program. In the context of this paper, a program is a set of modules designed to carry out an operational mission, such as the computer programs used in NTDS aboard an aircraft carrier.

APPROACH

Ideally, the first step in software quality management would be the establishment of quality specifications which would be used during the testing period to determine the acceptability of the software product for its intended use. Unfortunately, the use of software specifications which pertain to error, as opposed to performance characteristics, is not widespread. Since there is normally no baseline against which test results can be compared, a problem arises concerning the choice of criteria for determining the acceptability of a software product. The approach used in this model is to monitor the occurrence

of software errors and to forecast future numbers of cumulative detected and corrected errors in order to:

(1) Identify the trade-off function between error reduction and the cost of error reduction, where cost may be measured in terms of calendar time, computer time,or manpower required.

(2) Provide a quantitative basis for accepting or rejecting software during functional testing.

(3) Provide a quantitative basis for deciding whether additional testing is warranted based on the relationship between incremental error reduction and incremental cost.

Quantitative measures of software quality which can be applied to the above are:

- cumulative number of detected or corrected errors as a function of time,
- rate of error detection or correction; this is the first derivative of the preceding function,
- number of errors that has been detected but not corrected after a specified time,
- time required to detect a specified number of cumulative errors,
- time required to correct a specified number of cumulative detected errors,
- time required to correct a specified number of detected but uncorrected errors.

In the above measures, the variable time can be either calendar time or software test time. Also, the measures apply to both historical and forecasted error values.

Unlike hardware which wears out or deteriorates with time, software should improve with time as more of the latent errors are detected and corrected.

However, there are exceptions to this general characteristic. When an error is removed, it is possible that one or more new errors will be introduced. Also, most operational software is subject to modification as the result of application changes or design improvements. Consequentially, the time series of error counts over equal time intervals will not necessarily be monotonically decreasing, and the time between error occurrences will not necessarily be monotonically increasing, over the life cycle of the software. However, the trend of these series will be decreasing and increasing, respectively, when observed over an extended time period [6].

It is important to recognize that, with exceptions occurring in trivial programs, software is seldom free of errors. Errors may reside undetected in software for many years until a particular set of input data causes a previously untraversed module path to be executed [4].

APPLICABILITY

Software Testing.

This model is applicable to the analysis of software errors which are detected during the functional test phase of software testing. It is not applicable to the detailed and highly individualistic test procedures employed by the programmer prior to the conduct of functional tests. In particular, we are concerned with tests which are made after the individual modules are linked together as a program. The reason for the distinction is that programmer debugging procedures are highly individualized [1]. Also, the selection of test procedures and test data may be a deterministic process and, in general, be less suitable for a probabilistic analysis than functional testing. Functional testing is designed to test the ability of the program to produce desired

outputs with a given sequence of inputs. Inputs may consist of console operator actions; inputs from radar, magnetic tape or other sensors and devices; or a script of inputs recorded on magnetic tape which simulates console operator input actions. Any input errors are not counted as software errors.

Software Errors.

Software errors can be classified according to the programming and hardware characteristics of the error, such as an incorrect memory-to-memory transfer caused by an error in addressing. In addition, errors can be classified according to their effect on mission success. Both types of classification are useful. However, the latter classification is much more difficult to make than the former because it is necessary to associate three items of information:

(1) the manifestation of the error (incorrect target symbol on the display console);

(2) the effect on the mission (failure to identify a hostile target); and

(3) the programming cause of the error (incorrect interpretation by the program of an operator hostile target input).

Although the variability among error counts which are used to measure software quality could probably be decreased by classifying and counting errors by category, the resulting sample sizes would be significantly reduced. Consequently, as a practical matter, only a limited number of categories can be used for error classification. In NTDS software error reporting, errors are classified according to the effect on the mission as follows:

- High. The computer will stop if this type of error occurs. Example: attempt to address data outside the memory address range of the computer.

- Medium. This type of error will cause a degradation in system performance. Example: target position is not updated with sufficient frequency.
- Low. This type of error will be distracting or annoying but will not normally result in degraded performance, although lowered performance could result if the operator is unable to cope with the problem. Example: the programmed refresh rate of the display console is low and causes fading of symbol displays.

In order to achieve maximum sample size for evaluating the forecasting methodology used in the model, errors were not separated according to the classification given above. However, when errors are separated by category, the same forecasting methodology is used for each forecast. Forecasting accuracy will be affected by categorization because sample size and variability among error counts are reduced.

ASSUMPTIONS

It is assumed that the number of errors which is detected during a time interval and the collection of error counts over a series of time intervals are modelled by a random variable and a stochastic process, respectively. Errors which are repeated as a result of repeating the execution of the module under identical test conditions, and without having corrected the error, are not counted. Only "new" errors, which occur for the first time (execution), are counted.

Functional testing is not entirely probabilistic because test plans and procedures are structured to a certain extent and are not selected at random. Although the types of tests, test sequence and test data are not

randomly selected, variables such as time to detect, or correct, an error, and number of detections or corrections per time interval may be modelled as random variables.

The probability of detecting an error is a function of type of test plan, characteristics of input data, and number and locations of errors in a module. Prior to the selection of a test plan and input data, all errors (1 through 7 in Set A, Figure 1) have equal probability of detection because there is usually no prior knowledge concerning the presence of errors or the probability of error detection. Once the next test is selected, those errors falling outside the domain of the test (Errors 6 and 7 in Set B) now have zero probability of detection. Those errors falling within the domain of the test (Errors 1 through 5 in Set C) now have non-zero probability of detection. Once the input data are selected, the combination of input data and test plan leads to the following assumed situation:

- Error 1 in the set affected by the Input Data A has a probability of one of detection. Errors 2 and 3 in Set D now have zero probability of detection in this test because the detection of Error 1 will halt the computer. The future probability of detection of Errors 2 and 3 may be dependent upon the previous detection and correction of Error 1, i.e. the detection and correction of one error may allow another error to be detected because the path to the second error is no longer blocked by the first error.
- Errors 4 and 5 in Set E are not affected by Input Data A and have zero probability of detection.

Assuming that Error 1 is corrected and the test repeated, it may now be possible to detect Errors 2 or 3, if the detection of one or both depends

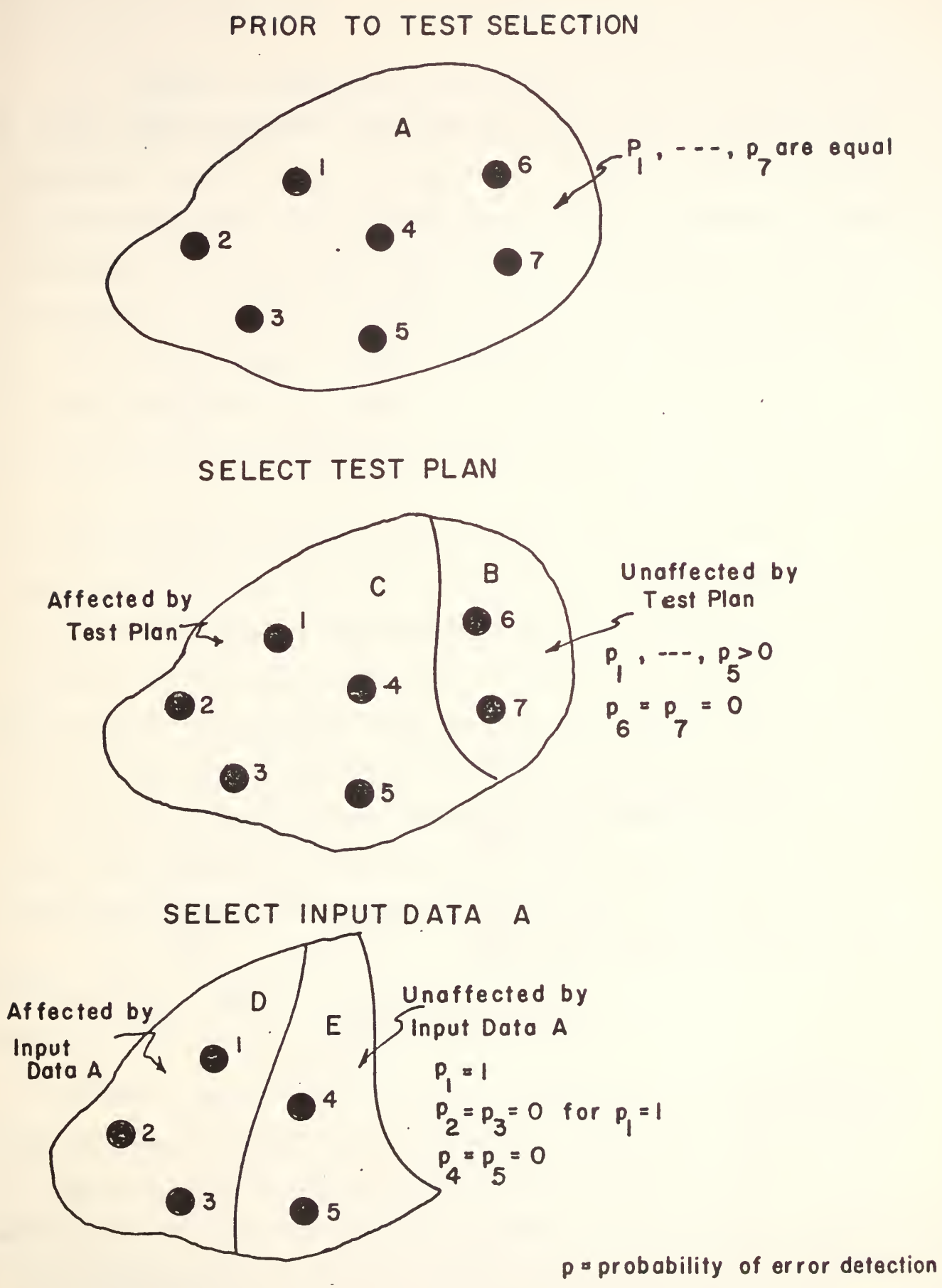

Figure 1. Error, test, and input data relationships.

upon Input Data A and the prior detection and correction of Error 1 (Figure 2). This is an example of the detection of an error being dependent on the detection and correction of another error. Conversely, the detection of Errors 2 and 3 may be independent of the use of Input Data A and the prior detection and correction of Error 1. This is an example of the detection of an error being independent of the prior detection and correction of another error.

Once a second Input Data Set B is chosen, and assuming that Errors 2 and 3 have not been detected, a new set of errors, Set F, is affected and Set G is not affected (Figure 2).

It should be stressed that the above probabilities and dependencies (if any) among errors are unknown prior to the execution of the tests. This information can be obtained only after detailed post mortem analysis of the tests. Generalizations concerning error dependencies are difficult to make due to the great variety of module structures. However, the probability seems low of having many situations in which errors are located in a module in such a way that the detection of Error 2 depends simultaneously on the removal of Error 1 and the use of the same test and input data which was used in the detection of Error 1. This reasoning leads to the assumption of independence among errors in the model formulation.

MODEL FORMULATION

A major objective of the model is to forecast the mean number of cumulative errors for some future time T, assuming that the forecast is made at time t, $t < T$, and observations have been made of the number of errors which have occurred in intervals of unit length, designated by the index i, from interval 1 through t. Due to the characteristics of the data, a calendar time scale

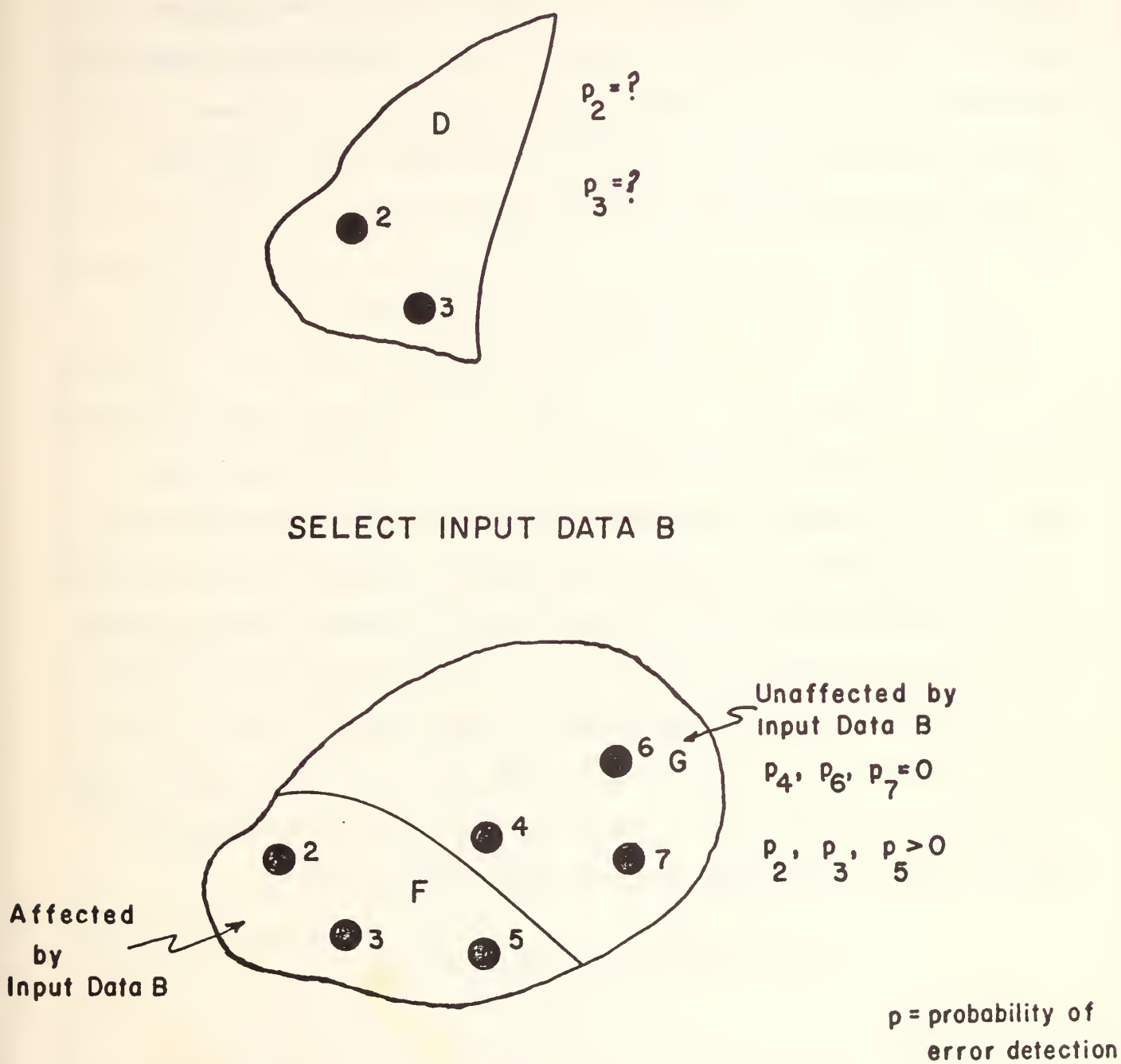

Figure 2. Error, test, and input data relationships (continued).

is used. Ideally, error counts should be made with respect to computer operating time intervals. However, the available data contains counts per calendar time interval. In order to make the count meaningful with respect to the exposure of a module to testing, a count interval of one week was chosen because, with respect to the data used in the analysis, modules are tested for approximately equal computer time durations each week.

Error Detection.

The actual number of errors detected during interval i is denoted by x_i and the estimate of the number of errors detected in interval i is denoted by m_i. It is assumed that: (1) in accordance with arguments presented earlier, the number of errors detected in each time interval is independent of the number of errors detected in any other time interval; (2) the detected error counts have a probability density function of the same form in each time interval but with different means; and (3) the mean number of detected errors decreases from interval to interval as a result of the continuing detection and correction of original errors. In addition, it is assumed that the rate of error detection in an interval is proportional to the number of errors in the interval. Specifically, the detected error process is assumed to be a non-homogeneous Poisson process with an exponentially decaying intensity function

$$d(i) = \alpha \exp(-\beta i), \qquad \alpha > 0, \quad \beta > 0; \tag{1}$$

mean value function

$$D(i) = (\alpha/\beta)[1-\exp(-\beta i)]; \tag{2}$$

and mean number of errors in each interval i equal to

$$m_i = (\alpha/\beta)[\exp(-\beta(i-1)) - \exp(-\beta i)]. \quad (3)$$

The time estimated to detect a cumulative number of errors D is derived from (2) as

$$i_d = \{\log[\alpha/(\alpha-\beta D)]\}/\beta. \quad (4)$$

The detection rate of errors is given by (1) and the time estimated for the detection rate to reach the value d is derived from (1) as

$$i'_d = [\log(\alpha/d)]/\beta. \quad (5)$$

Error Correction.

In addition to detected errors, the cumulative mean of which is given by (2), the software quality control function is also concerned with the correction of detected errors. Assuming that resources are committed to the correction of errors in proportion to number of errors detected, the cumulative mean corrected error function $C(i)$ will have the same form as (2) but will lag (2) by Δi. This is the time estimated to correct a number of errors equal to $D(i) - C(i)$. Thus, for $i \geq \Delta i$,

$$C(i) = D(i-\Delta i) = (\alpha/\beta)[1-\exp(-\beta(i-\Delta i))]. \quad (6)$$

The lag Δi can be estimated by finding Δi such that the relationship

$$C(t) = D(i-\Delta i) \quad (7)$$

is satisfied from the empirical data, where t is the time of making a forecast. These relationships are shown in Figure 3. The time estimated to correct a cumulative number of errors C is derived from (6) as

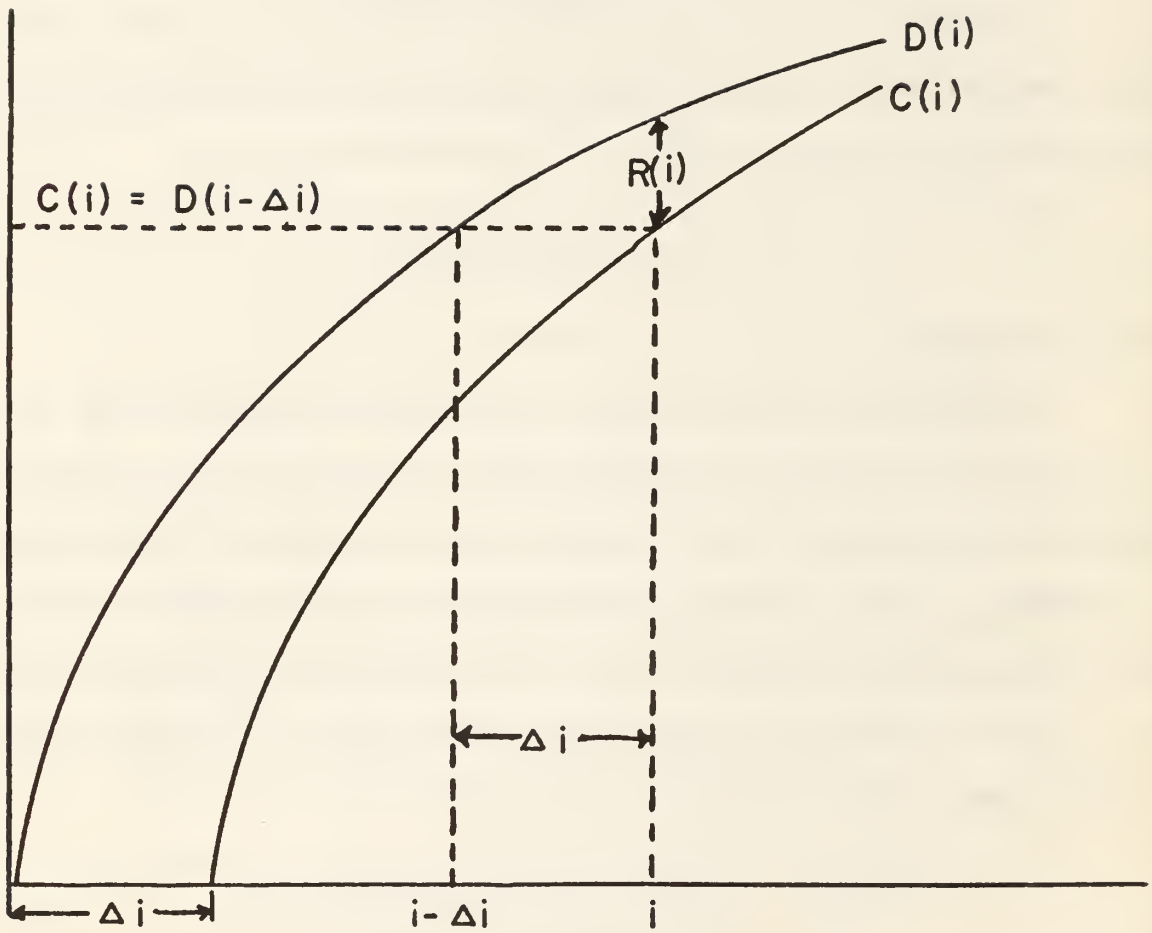

Figure 3. Relationship between cumulative detected D(i) and corrected C(i) errors.

$$i_c = \Delta i + \{\log[\alpha/(\alpha-\beta C)]\}/\beta. \quad (8)$$

The correction rate of errors is derived from (6) as

$$c(i) = \alpha \exp[-\beta(i-\Delta i)], \quad (9)$$

for $i \geq \Delta i$. The time estimated for the correction rate to reach the value c is derived from (9) as

$$i'_c = \Delta i + [\log(\alpha/c)]/\beta. \quad (10)$$

Difference Between Detected and Corrected Errors.

The number of detected errors which has not been corrected after time i is derived from (2) and (6) as

$$R(i) = D(i) - C(i) = (\alpha/\beta)[\exp(-\beta i)][\exp(\beta\Delta i)-1], \quad (11)$$

for $i \geq \Delta i$. This relationship is shown in Figure 3. For a given value of $R(i)$ existing at time i, the time (Δi) estimated to correct R errors is derived from (11) as

$$\Delta i_r = [\log(R\beta \exp(\beta i)/\alpha+1)]/\beta. \quad (12)$$

The time estimated to reach a given value of R is derived from (11) as

$$i_r = \{\log[\alpha(\exp(\beta\Delta i)-1)/R\beta]\}/\beta. \quad (13)$$

All of the above expressions except (3), which is used for parameter estimation as described in the next section, may be used as quantitative aids in software quality control and assurance functions.

ERROR FORECASTING METHODOLOGY

This section will describe the methodology used for estimating the error detection and correction function parameters α and β. Once these parameters have been estimated, forecasts are made of cumulative detected and corrected errors in the forecast interval $t+1$ through T. With forecasts available, the type of analyses described in the previous section can be made.

Parameter estimates are made by using the criteria of maximum likelihood and weighted least squares. The first approach was to use the method of maximum likelihood only and to use all error count observations x_i in the interval 1 through t. It was found that forecasting accuracy, as computed by the sum of squared deviations in the interval $t+1$ through T, was unacceptable for error prediction purposes. This result is caused by differences between the actual and model processes which occur over an extended period of time. That is, α and β appear to be constant only over restricted time intervals, and vary when the observation and forecast intervals are long. A further problem is that the time of error observation is not recorded on NTDS software trouble reports. The date information provided on the trouble report is date of report preparation rather than date of observation. In many cases the two dates are the same; in other cases there is a time lag between observation and report preparation and, hence, a difference in dates. The extent of time recording discrepancies cannot be obtained from the data originator. This source of error does not appear to be a major contributor to forecast inaccuracies. Since, in general, software error data is very difficult to obtain, whereas the supply of NTDS data is plentiful, the approach which has been taken is that the model must be capable of utilizing a certain amount of imprecise data and still provide reasonable forecasting accuracy.

Since the error process apparently changes over time, recent observations are generally more useful than earlier observations, although this will not be the case in every forecasting situation. Therefore, when selecting a forecasting methodology, methods which provide for unequal representation of error counts in the forecast are considered in addition to a method which uses all counts on an equal basis. In order to accomplish this objective, it was necessary to develop criteria for determining:

(1) to what extent historical observations would be considered in forecasting,

(2) how much of the historical time record to include when estimating α and β,

Three methods were developed:

(1) All of the error counts in intervals from 1 through t are used.

(2) None of the error counts in intervals from 1 through $s-1$ are used, where s is an index, with unit increment, $2 \leq s \leq t$; all of the counts in intervals from s through t are used.

(3) The cumulative error count from intervals 1 through $s-1$ is used; individual error counts in intervals from s through t are used.

Method (1) is appropriate if changes in error count from interval 1 through t, for all intervals, are representative of the future ability to detect errors. Method (2) is appropriate if the most recent error count changes from interval s through t are representative of the future ability to detect errors. Finally, Method (3) is intermediate to (1) and (2), and is appropriate if the individual error count changes from interval 1 through $s-1$ are not representative of the future ability to detect errors, but the total error count from 1 through $s-1$ and the changes in error count from s through t are representative.

Method (1) involves applying the method of maximum likelihood to all error counts x_i from interval 1 through t, Methods (2) and (3) require a criterion for selecting s. This was accomplished by first estimating α and β (by the method of maximum likelihood), for each value of s from 2 through t, and then computing the sum of weighted squared deviations SD_w between error estimates m_i and actual errors x_i from interval 1 through t, where the intervals are of equal length and the summation for each value of s is computed over the same number of intervals. The best value of s in Methods (2) and (3) is determined by choosing the value s, and corresponding positive values of α and β, that produce the minimum value of SD_w. The three methods can be evaluated by comparing values of SD computed from unweighted squared deviations between forecasted errors and actual errors in the interval t+1 through T.

Weighted Least Squares Criterion.

The variance of the deviation e_i between estimated errors m_i and actual errors x_i, when zero bias is assumed, is equal to the variance of m_i, as shown by

$$\mathrm{Var}(e_i) = E[(m_i - x_i)^2] = E(m_i^2) - x_i^2 \tag{14}$$

$$\mathrm{Var}(m_i) = E(m_i^2) - [E(m_i)]^2 = E(m_i^2) - x_i^2. \tag{15}$$

This variance is also given by

$$\mathrm{Var}(e_i) = E(e_i^2) - [E(e_i)]^2 = E(e_i^2), \tag{16}$$

assuming that $E(e_i) = 0$. Also, since the process is assumed to be Poisson, the mean and variance are equal. Combining this fact with (3), (14), (15) and (16), we have

$$\mathrm{Var}(e_i) = \mathrm{Var}(m_i) = E(e_i^2) = (\alpha/\beta)[\exp(-\beta i)][1-\exp(-\beta)]. \tag{17}$$

In order that $Var(e_i) = E(e_i^2)$ be constant, as required by the method of least squares, (17) must be multiplied by the term $\exp(\beta i)$ in order to eliminate its time-varying term. Thus,

$$Var(e_i') = E[(e_i')^2] = \exp(\beta i)Var(e_i) = \exp(\beta i)E(e_i^2), \qquad (18)$$

where e_i' is a constant variance deviation term. Since $E(e_i^2) = E[(m_i - x_i)^2]$, this fact combined with (18) gives

$$E[(e_i')^2] = \exp(\beta i)E(e_i^2) = \exp(\beta i)E\{\{(\alpha/\beta)[\exp(-\beta i)][1-\exp(-\beta)]-x_i\}^2\}. \qquad (19)$$

Thus, in order to select the best values α^* and β^* in Methods (2) and (3), by the least squares criterion, involving the minimization of $\sum_i (e_i')^2$, we find $s = s^*$ such that

$$SD_w = \sum_{i=1}^{i=t} \exp(\beta i)\{(\alpha/\beta)[\exp(-\beta i)][1-\exp(-\beta)]-x_i\}^2 \qquad (20)$$

is minimized. In order to compare Methods (1), (2) and (3), SD (unweighted) is computed over the interval from $t+1$ through T. The method which produces the lowest values of SD for positive values of α and β is the preferred method.

Maximum Likelihood Estimation of Parameters.

The parameters α and β are estimated by the method of maximum likelihood. For Method (3), the estimate is made with respect to a starting interval s for making observations of the number of errors per time interval. The relationship between time intervals and error counts is shown in Figure 4.

The development of the likelihood function $L(\alpha,\beta,s)$, where the constant factors $x!$ are ignored in the density functions, for estimating parameters for Method (3) follows. This is also the formulation for Method (1) when $s=1$.

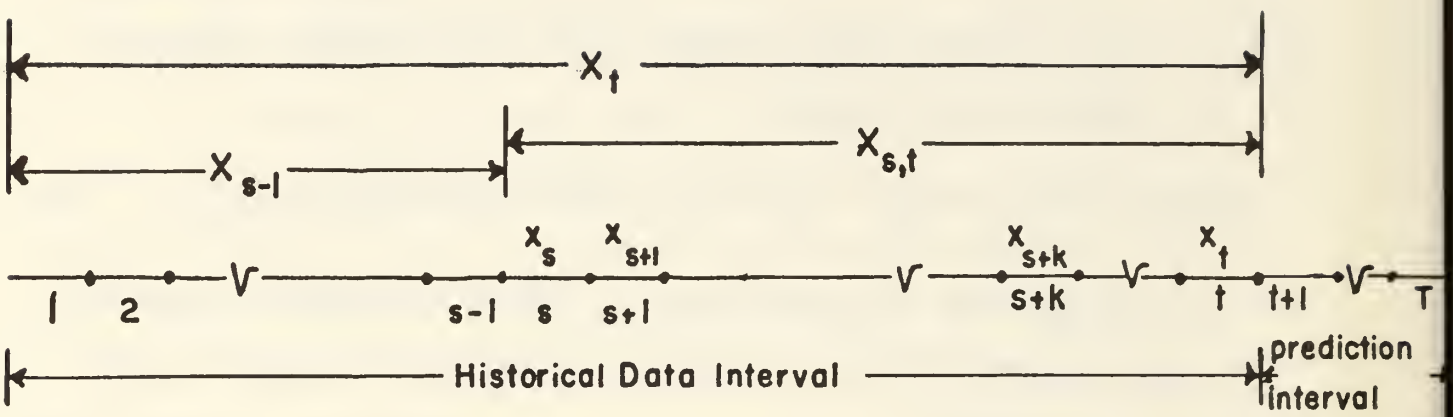

Figure 4. Relationship between time intervals and error counts.

$$L(\alpha,\beta,s) = [(\exp(-M_{s-1}))(M_{s-1}^{X_{s-1}})][(\exp(-m_s))(m_s^{x_s})][(\exp(-m_{s+1}))(m_{s+1}^{x_{s+1}})] \ldots$$

$$[(\exp(-m_{s+k}))(m_{s+k}^{x_{s+k}})] \ldots [(\exp(-m_t))(m_t^{x_t})] \qquad (21)$$

where M_{s-1} is the mean number of errors in the interval 1 through s-1, m_s, m_{s+1}, m_{s+k} and m_t are the mean number of errors in intervals s, s+1, s+k and t, respectively, and X_{s-1}, x_s, x_{s+1}, x_{s+k} and x_t are the corresponding numbers of errors. The mean number of errors, based on the assumption of a Poisson distribution in each interval, is as follows:

$$M_{s-1} = (\alpha/\beta)[\exp(0)-\exp(-(s-1)\beta)] = (\alpha/\beta)[1-\exp(-(s-1)\beta)]$$

$$m_s = (\alpha/\beta)[\exp(-(s-1)\beta)-\exp(-s\beta)] = (\alpha/\beta)[\exp(-(s-1)\beta)][1-\exp(-\beta)]$$

$$m_{s+1} = (\alpha/\beta)[\exp(-s\beta)-\exp(-(s+1)\beta)] = (\alpha/\beta)[\exp(-s\beta)][1-\exp(-\beta)]$$

$$\vdots$$

$$m_{s+k} = (\alpha/\beta)[\exp(-(s+k-1)\beta)-\exp(-(s+k)\beta)] = (\alpha/\beta)[\exp(-(s+k-1)\beta)][1-\exp(-\beta)]$$

$$\vdots$$

$$m_t = (\alpha/\beta)[\exp(-(t-1)\beta)-\exp(-t\beta)] = (\alpha/\beta)[\exp(-(t-1)\beta)][1-\exp(-\beta)]$$

Substituting the foregoing in (21) and taking the natural logarithm, gives

$$\log L = -(M_{s-1}+m_s+m_{s+1}+\ldots+m_{s+k}+\ldots+m_t) + X_{s-1} \log m_{0,s-1}$$

$$+ x_s \log m_s + x_{s+1} \log m_{s+1} + \ldots + x_{s+k} \log m_{s+k} + \ldots + x_t \log m_t$$

$$= (\alpha/\beta)[\exp(-\beta t)-1] + X_{s-1}[\log(\alpha/\beta) + \log(1-\exp(-(s-1)\beta))] +$$

$$x_s[\log(\alpha/\beta)-(s-1)\beta+\log(1-\exp(-\beta))]+x_{s+1}[\log(\alpha/\beta)-s\beta+\log(1-\exp(-\beta))]+\ldots+$$

$$x_{s+k}[\log(\alpha/\beta)-(s+k-1)\beta+\log(1-\exp(-\beta))]+\ldots+x_t[\log(\alpha/\beta)-(t-1)\beta+\log(1-\exp(-\beta))].$$

Taking the partial derivatives $\partial \log L/\partial\alpha$ and $\partial \log L/\partial\beta$ and setting these equal to zero, we have

$$\hat{(\alpha/\beta)} = X_t/[1-\exp(-\beta t)] \tag{22}$$

and $$(s-1)(X_{s-1})/[\exp((s-1)\beta)-1]+X_{s,t}/[\exp(\beta)-1]-tX_t/[\exp(\beta t)-1] = \sum_{k=0}^{t-s}(s+k-1)x_{s+k}, \tag{23}$$

where $X_{s,t}$ and X_t are the number of errors detected in the interval s to t and 1 to t, respectively. When $s = 1$, (23) reduces to

$$1/[\exp(\beta)-1] - t/[\exp(\beta t)-1] = (\sum_{k=0}^{t-1} kx_{k+1})/X_t. \tag{24}$$

This is the result that would be obtained if the method of maximum likelihood was applied to $x_1, x_2, x_3, \ldots, x_t$ (the individual error counts in every interval from 1 through t). It should be noted that when $s = 1$ or $s = 2$ is substituted in (23), the same result (24) is obtained.

In order to estimate α and β in accordance with Method (2) all error counts in the interval 1 through $s-1$ are ignored. This is accomplished by substituting $t - s + 1$ for t in (24) in order to account for the fact that there are only $t - s + 1$ intervals to consider, when the first $s-1$ intervals are ignored. In addition, the subscript of x in the summation of (24) must be adjusted to make x_s the first error count. The resulting expression is

$$1/[\exp(\beta)-1] - (t-s+1)/[\exp(\beta(t-s+1))-1] = (\sum_{k=0}^{t-s} kx_{s+k})/X_t. \tag{25}$$

RESULTS

A limited test of the validity of the model was made by evaluating the forecasting accuracy against software error data from an NTDS module. This module

had a total of 160 detected errors over a period of 132 weeks. The first 20 weeks, involving 30 detected errors, was used to make a variety of forecasts, using the model equations and forecasting methodology previously described. Forecasts of cumulative detected and corrected errors were made for weeks 21-30, 21-40 and 21-50 and compared with actual values. Forecasts were also made of the time required to detect or correct a specified number of errors. In addition, an evaluation was made of the accuracy of the three forecasting methodologies: Methods (1), (2) and (3). The composition of 30 errors in the observation period (weeks 1-20) is as follows:

Source		Severity	
Production	30%	Low	3%
Test	70	Medium	37
	100	High	60
			100

The composition of the 53 errors during the forecast period (weeks 21-50) is as follows:

Errors: Stages		Errors: Severity	
Production	8%	Low	6%
Test	79	Medium	31
User	13	High	63
	100		100

Production errors are those errors detected during the time the software contractor has cognizance of the software. Test errors are those errors detected by the customer during functional test, after the software has been delivered by the contractor to the customer. User errors are those errors detected by the customer after functional testing has been completed and the software has been put into operational use. The High, Medium and Low categories are the error severity classifications used in NTDS trouble reports, as previously

described. Although forecasts can be made by category, this was not done in the analysis being discussed because, for the initial evaluation, it was desired to use the maximum amount of data.

In general, the errors used in this analysis were detected under conditions which closely approximate a functional testing environment--the type of testing for which the model is applicable. Also, the composition of errors in the observation period is reasonably representative of the composition of errors in the forecast periods. Descriptions follow of the various analyses which were performed.

Forecast Error as a Function of s.

This analysis was performed in order to determine whether forecast error varies as a function of s, the first interval where individual detected software error counts are used for forecasting in Methods (2) and (3). Figure 5 shows forecast error as a function of s for three forecast periods for Method (1), $s = 1$, and Method (3), $s = 2 - 15$. Method (2) is not shown because a number of negative β's were generated as s was varied. Negative values of β have no meaning because the model is based on the assumption that the intensity function is a decreasing function, at least over an extended period of time, and that a counter trend would be attributable to errors in observation, or to the introduction of new software errors as a result of attempting to correct another software error. Method (1) is included on Figure 5 because it corresponds to $s = 1$ in Method (3). It is seen that, for the module tested, forecast error varies considerably with s for the more distant forecast periods and that using the maximum amount of data does not provide the greatest accuracy. Whereas forecast error is relatively insensitive to s for short range forecasting, it is very sensitive to s (decreases

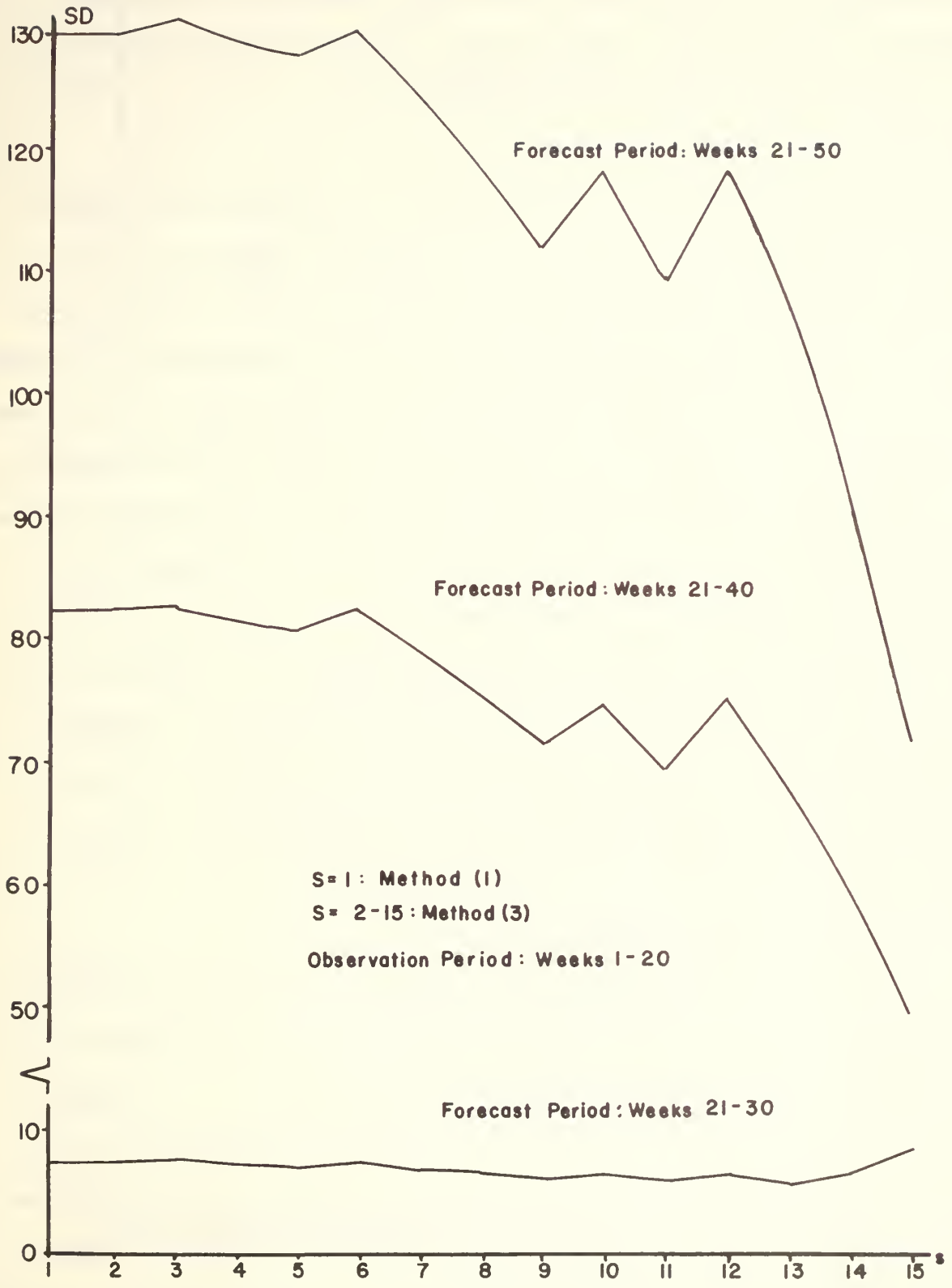

Figure 5. Sum of forecast deviations squared SD vs s for detected errors

with increasing s) for long range forecasting. Apparently, the process changes over time, and recent software error counts are more representative of the changing process than are early error counts.

Forecast Error and Weighted Least Squares Criterion.

It was of interest to test the validity of the weighted least squares criterion. This criterion is used to select particular values of α and β from the set of paired values obtained from maximum likelihood parameter estimates of the detected software error process. Validity was tested by determining how well forecast error (as measured by the sum of unweighted deviations squared in the forecast period) associates with the sum of weighted deviations squared in the observation period. The values shown in Figure 6 were obtained by using various values of s (2-15) for Method (3). The positive association is reasonably good. Also shown are plots of forecast error versus the sum of unweighted squared deviations in the observation period. It is seen that this association is basically negative. Hence, as would be expected, a weighted least squares criterion is superior to an unweighted least squares criterion for the type of model being analyzed (variance of error count decreases exponentially with time).

Forecast Error and Forecast Methodology.

Cumulative forecast error is plotted against forecast week for detected software errors for the three methods in Figure 7. In the case of Methods (2) and (3), the error function corresponding to the best value of s is plotted. This is the value which produces minimum error during the observation period, weeks 1-20. The three methods are also compared in Figure 8, where cumulative actual and forecasted detected (using Equation 2) errors are plotted. Again, the best values of s are used for Methods (2) and (3). In general, fore-

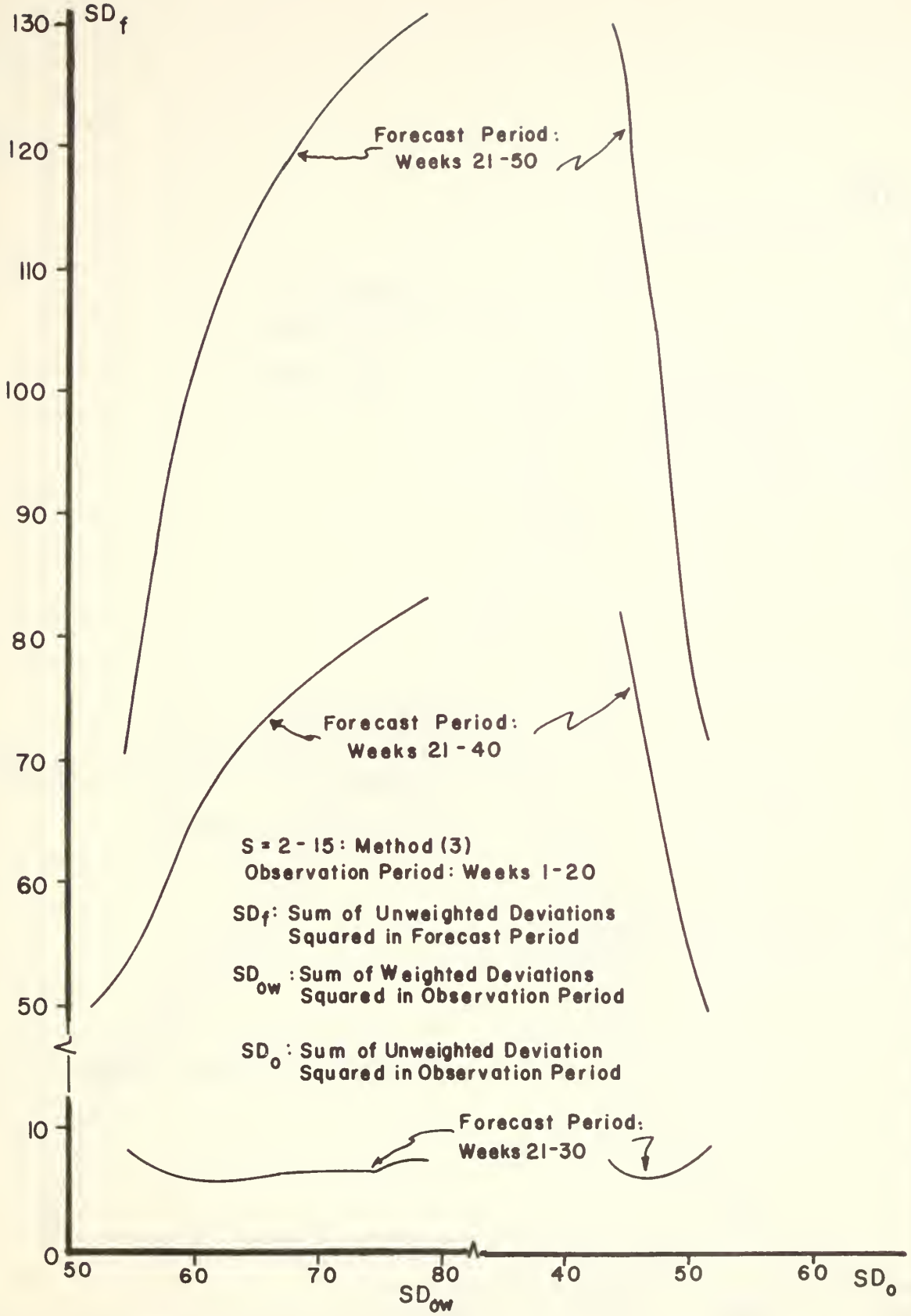

Figure 6. Forecast deviations SD vs weighted observation deviations SD_{ow} and vs unweighted observation deviations SD_o for detected errors.

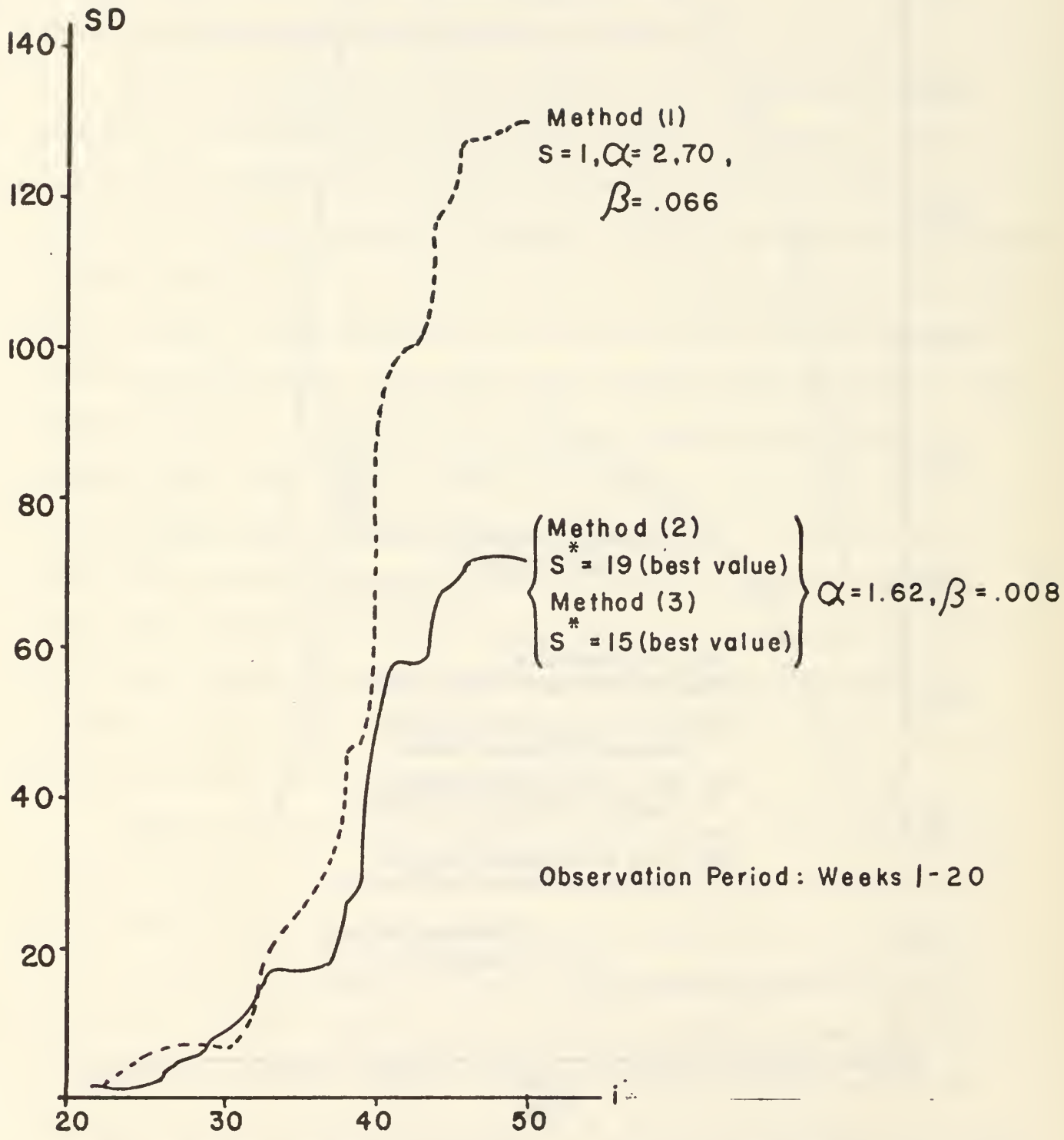

Figure 7. Sum of forecast deviations squared SD vs forecast week i for detected errors.

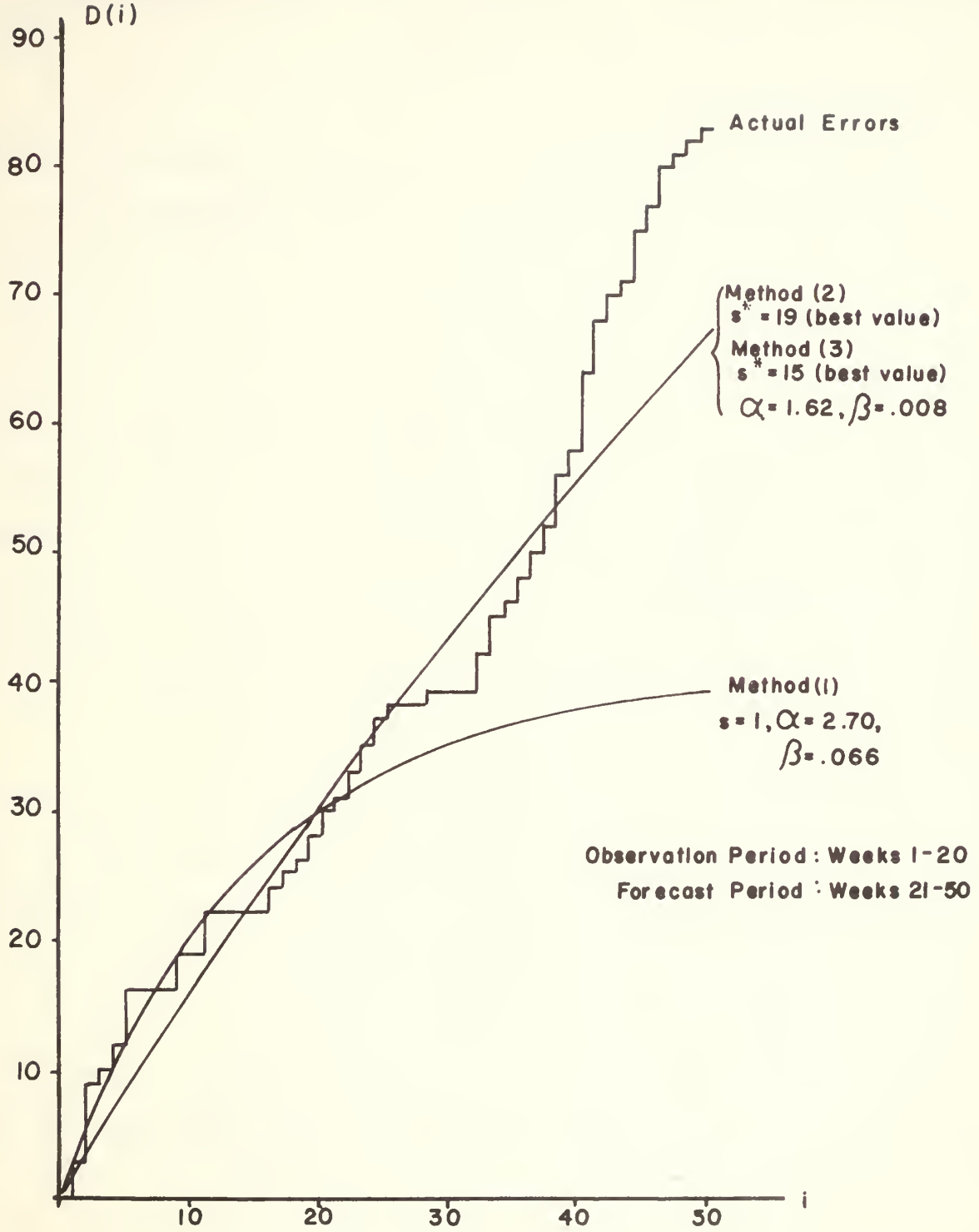

Figure 8. Cumulative actual and forecasted detected errors D(i) vs week i.

casting accuracy is greater for Methods (2) and (3), because recent software error observations are more representative of the changing software error detection process than are early observations.

Detected and Corrected Errors.

A test was made of the validity of the forecasted cumulative corrected error function, as given by equation (6). This was accomplished by first obtaining an estimate of Δi, the lag between error detection and correction, by using the empirical data and equation (7). Then, using the best values of α and β, corresponding to the best value of s, equation (6) was used to forecast cumulative corrected errors and compared to actual cumulative errors in Figure 9. In addition, in order to show the contrast between detected and corrected errors, cumulative forecasted and actual detected errors are plotted in Figure 9. As would be expected, the accuracy of corrected error forecasts is less than the accuracy of detected error forecasts because, in addition to estimates of α and β, it is also necessary to make an estimate of Δi.

Time to Detect Errors.

The forecasted time i_d to detect a specified number of cumulative errors, as given by equation (4), was computed for five actual values of $D(i)$ and compared with the actual detect time i in Table I. Forecast accuracy is good for short range forecasts but decreases for long range forecasts.

Time to Correct Cumulative Errors.

The forecasted time i_c to correct a specified number of cumulative errors, as given by equation (8), was computed for five actual values of $C(i)$ and compared with the actual time to correct i in Table I.

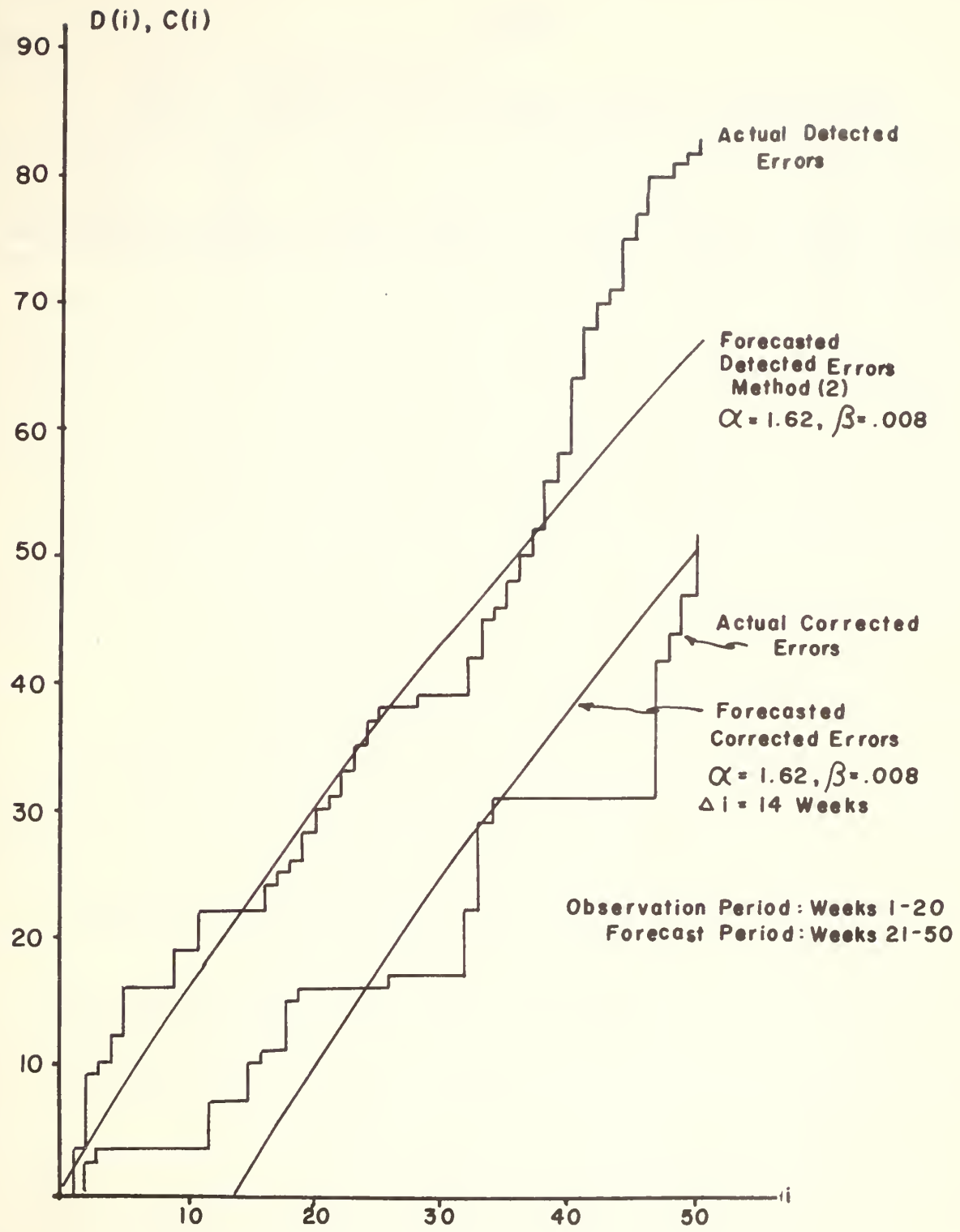

Figure 9. Cumulative actual and forecasted detected D(i) and corrected C(i) errors vs week i.

TABLE I

COMPARISON OF ACTUAL AND FORECASTED ERROR VALUES

i	$D(i)$	i_d	$C(i)$	i_c	$R(i)$		Δi_r	
(Actual)	(Actual)	(Forecast)	(Actual)	(Forecast)	(Actual)	(Forecast)	(Actual)	(Forecast)
30	39	26.7	17	25.0	22	18.9	16.7	16.1
35	48	33.7	31	34.7	17	18.2	14.2	13.1
40	64	47.2	31	34.7	33	17.6	18.4	25.2
45	77	59.3	31	34.7	46	16.9	20.3	35.0
50	83	65.3	52	50.9	31	16.2	22.5	25.5

i actual time in weeks

$D(i)$ actual number of cumulative detected errors

i_d forecasted time to detect $D(i)$ number of cumulative errors (in weeks)

$C(i)$ actual number of cumulative corrected errors

i_c forecasted time to correct $C(i)$ number of cumulative errors (in weeks)

$R(i)$ actual and forecasted number of detected but uncorrected errors

Δi_r actual and forecasted time required to correct $R(i)$ number of errors

Δ_i lag in correcting errors = 14 weeks

$\alpha = 1.62, \quad \beta = .008$

Number of Detected but Uncorrected Errors.

The number of detected but, as yet, uncorrected (remaining) errors R(i), which would exist at time i, was forecasted for five values of i, by using equation (11), and compared with the actual number of such errors in Table I. Forecast accuracy is good for short range forecasts but decreases for long range forecasts. It is to be expected that the accuracy of R(i) forecasts will not be as great as the accuracy of D(i) and C(i) forecasts, because R(i) is a function of both D(i) and C(i).

Time to Correct Remaining Errors.

The time Δi_r to correct a given number of detected but uncorrected errors was forecasted for five values of R(i), by using equation (12), and compared with the actual time in Table I.

CONCLUSIONS

Since only a single software module was analyzed, although one with a large number of reported software troubles, it would be inappropriate to generalize the results to the universe of NTDS, and certainly to that of other types of software. However, based on (unpublished) analysis by the author of approximately thirty NTDS modules, the great similarity in the characteristics (amplitude and shape) of the time series of detected errors among modules suggests the applicability of the model to NTDS software in general and, possibly, to other large scale software production activities. On the basis of limited experience, the forecasting accuracy of the various types of predictors seems adequate, and the predictors could be employed as decision aids in software testing management.

Additional areas for research are: (1) the validation of the model against a large number of NTDS modules, (2) validation of the model against other large scale software testing error data and (3) the collection and use of software testing resources (manpower, computers, money) utilization data, in conjunction with this model, for the development of functions which would indicate the trade-off between software quality improvement and resource expenditure.

ACKNOWLEDGMENTS

This work was sponsored by the Naval Electronics Laboratory Center, Fleet Material Support Office and the Office of Naval Research. The author is indebted to the Fleet Combat Direction Systems Support Activity for providing NTDS software trouble report data. The author wishes to thank Professor Donald Gaver of the Naval Postgraduate School for his many helpful comments.

REFERENCES

[1] F. T. Baker, "Chief Programmer Team Management of Production Programming," IBM Systems Journal, Vol. 11, No. 1, pp. 56-73 (1972).

[2] Myron Lipow, "Estimation of Software Package Residual Errors," TRW Systems Group, Report 2240.4B.72, November 1972.

[3] H. D. Mills, "On the Statistical Validation of Computer Programs," IBM Corporation Report FSC72-6015 (1972).

[4] G. J. Schick and R. W. Wolverton, "Assessment of Software Reliability," McDonnell-Douglas Astronautics Company Paper WD1872, August 1972.

[5] N. Schneidewind, "An Approach to Software Reliability Prediction and Quality Control," AFIPS Conference Proceedings, Vol. 41, Part II, Fall Joint Computer Conference, pp. 837-838 (1972).

[6] Martin L. Shooman, "Operational Testing and Software Reliability Estimation During Program Development," Record of IEEE Symposium on Computer Software Reliability, pp. 51-57 (1973).

INITIAL DISTRIBUTION LIST

	No. Copies
Dean of Research Code 023 Naval Postgraduate School Monterey, California 93940	2
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12
Library (Code 0212) Naval Postgraduate School Monterey, California 93940	2
Library (Code 55) Naval Postgraduate School Monterey, California 93940	2
W. R. Church Computer Center Naval Postgraduate School Monterey, California 93940	1
Computer Sciences Department, Code 5300 Naval Electronics Laboratory Center 271 Catalina Boulevard San Diego, California 92152	
Mr. A. E. Beutel	1
Mr. Joseph Dodds	1
Naval Electronics Laboratory Center Library 271 Catalina Boulevard San Diego, California 92152	1
Fleet Material Support Office Mechanicsburg, Pennsylvania 17055	
Commanding Officer	1
Mr. Edward Lukanuski	1
Commander Stephan Ruth (Code 0451) Department of Navy Supply Systems Command Washington, D. C. 20376	1
Stanford Research Institute Menlo Park, California 94025	
Mr. D. B. Parker	1
Mr. R. E. Keirstead	1

	No. Copies
Naval Air Development Center Warminster, Pennsylvania 18974	
Mr. H. Stuebing	1
Mr. R. Pariseau	1
Fleet Combat Direction Systems Support Activity 200 Catalina Boulevard San Diego, California 92147	
Commanding Officer	
Mr. M. Griswold	1
Mr. J. Hilliard	1
Mr. Marvin Denicoff Office of Naval Research Department of the Navy Arlington, Virginia 22217	1
Professor D. Gaver	1
Professor R. Butterworth	1
Professor G. Barksdale	1
Professor P. Lewis	1
Professor J. Esary	1
Professor R. Richards	1
Professor M. Thomas	1
Professor D. Williams	1
Professor A. McMasters	1
Professor G. Howard	1
Professor K. Marshall	1
Professor V. M. Powers	1
Professor U. Kodres	1
Professor N. Schneidewind	55